Projet transverse
Groupe 100

Conception d’un système de vidéosurveillance intelligente pour l’IMT



Sébastien ANGOT
Godefroy BRISEBARRE
Charlotte CADORET
Marc NAVATIER
David NICOLET-DIT-FELIX
Damien PINEAU



Remerciements

Nous tenons tout d'abord à remercier M. Ali Khalighi, notre tuteur de Projet Transverse, qui nous a accompagné pendant toute la durée de notre travail. Grâce à ses conseils, aussi bien en matière de gestion de projet que sur les domaines techniques abordés, nous avons pu avancer efficacement sur le projet. Il a également su nous soutenir lors des différentes phases du projet.

Nous souhaitons également remercier M. Pagé et toute l'équipe du CRI pour leur disponibilité et leur aide précieuse.

Nous remercions également deux personnes de l'entreprise MatrixVision, M. Hanu pour le support technique qu'il nous a fourni ainsi que M. Bonmart pour nous avoir proposé une offre de caméra dans nos moyens financiers.

Enfin, nous adressons nos sincères remerciements à M. Derrode, notre client, pour l'intérêt qu'il nous a porté lors de ce projet et la confiance qu'il nous a accordée pour le réaliser.



SOMMAIRE

Table des Figures

Introduction

Dans le cadre des Projets Transverses, nous avons choisi de nous intéresser au traitement d'image appliqué à la vidéosurveillance.

La surveillance, qui était autrefois accomplie uniquement par des agents de sécurité, a vu arriver la technologie comme une révolution. Alarmes, vidéosurveillance permettent aux agents d'être plus efficaces car ils visualisent directement de leur poste de sécurité tous les points clé des locaux à surveiller. Pourtant l'homme reste imparfait. En effet, qui pourrait rester devant des écrans de surveillance sans aucune faille ? C'est impossible. C'est là qu'apparaît la vidéosurveillance intelligente, elle permet d'aider les agents dans leur travail. Le programme peut détecter une intrusion à tout moment et en avertir l'agent qui prendra les mesures nécessaires. La surveillance en est rendue bien plus efficace et moins contraignante pour les agents.

Dans notre projet transverse, nous nous sommes penchés sur la conception d'un système de vidéosurveillance intelligente pour l'IMT (Institut Méditerranéen de Technologie dont fait partie l'Ecole Centrale Marseille). Nous avons dû évaluer les besoins du site en termes de nombre de caméras et d'emplacements de celles-ci. Ensuite, nous avons créé un programme de traitement d'image visant à détecter une anomalie et à en informer l'agent de surveillance. Notre système de vidéosurveillance assurera une meilleure efficacité que le système actuel et un plus grand confort pour les agents de surveillance.

I- Présentation du sujet

I.1- *Contexte global du projet*

L'École Centrale Marseille est située sur le technopôle de Château-Gombert. Cette école d'ingénieurs est également un centre de recherche de haut niveau, par conséquent, elle possède dans ses locaux du matériel de haute technologie et donc très onéreux. Il existe également toute une aile de l'école dédiée aux ressources informatiques, qui pourrait faire l'objet de convoitise : il y a de nombreux ordinateurs. De plus, le campus est un espace grand et ouvert sur l'espace public. Toute personne peut entrer à l'intérieur du campus à pied sans aucun contrôle. Il est donc nécessaire d'équiper l'établissement d'un système de vidéosurveillance efficace pour préserver les personnes appartenant à l'établissement ainsi que les locaux.

I.2- *Interprétation de la problématique*

La problématique de ce projet est de concevoir un système de vidéosurveillance intelligente pour surveiller les locaux de l'IMT (Institut Méditerranéen de Technologie). Pour mieux comprendre les enjeux de ce projet, nous avons réalisé un schéma QQOQCP donné ci-dessous.

		POURQUOI ?
Qui, pour qui ?	• Ecole Centrale Marseille • Agents de surveillance de l'IMT	• Meilleure fiabilité du système de surveillance • Faciliter le travail des agents de surveillance
Quoi, avec quoi ?	Système de vidéosurveillance intelligente	Caméras nécessaires pour un traitement de l'image
Où ?	Terrain qui est aujourd'hui propriété de l'IMT	Dans le futur l'Ecole Centrale devrait investir tous les locaux de l'IMT
Comment ?	• Programme informatique de traitement des images visant à reconnaître une situation anormale (intrusion) et à avertir les agents de sécurité	

Figure 1 : Tableau « QQOQCP »

Dans un premier temps, nous allons donc choisir le système de caméras à utiliser pour le projet, le nombre de caméras ainsi que leurs positions. Ensuite, il s'agira de concevoir un programme de traitement des images visant à reconnaître une situation anormale (intrusion) et à avertir les agents de sécurité.

I.3- *Objectifs du projet*

Après une documentation générale sur le sujet de la vidéosurveillance intelligente, et notamment la détection de scène d'intrusion, nous nous sommes rendu compte qu'il serait probablement difficile dans le temps qui nous est imparti de faire un catalogue des scènes possibles et de les intégrer au programme de détection. Cette partie relève d'un travail de sociologie.

Nous allons donc, en premier lieu, nous concentrer sur le choix du système de caméras à utiliser pour le projet de vidéosurveillance de l'IMT en se basant notamment sur le retour d'expérience des agents de sécurité du site. Ensuite, nous devrons explorer les différents traitements d'images connus pour la détection des mouvements. Enfin, nous nous appliquerons à concevoir un programme de traitement d'images dans le but d'effectuer cette détection de mouvement sur des flux vidéo.

II- Organisation du projet

II.1- *Synthèse des outils de gestion de projet*

Pour cerner la problématique du projet, nous avons d'abord élaboré un schéma QQOQCP. Cela nous a aidé à comprendre les enjeux de cette problématique, tant sur les points techniques à traiter que les utilisateurs du système. (Le QQOQCP est donné ci-dessus dans la partie *Interprétation de la problématique*)

Pour définir toutes les fonctions constructives de notre système de vidéosurveillance intelligente à partir des fonctions de service que doit produire le système, nous avons rédigé un diagramme FAST (Functional Analysis System Technic) présenté ci-dessous.

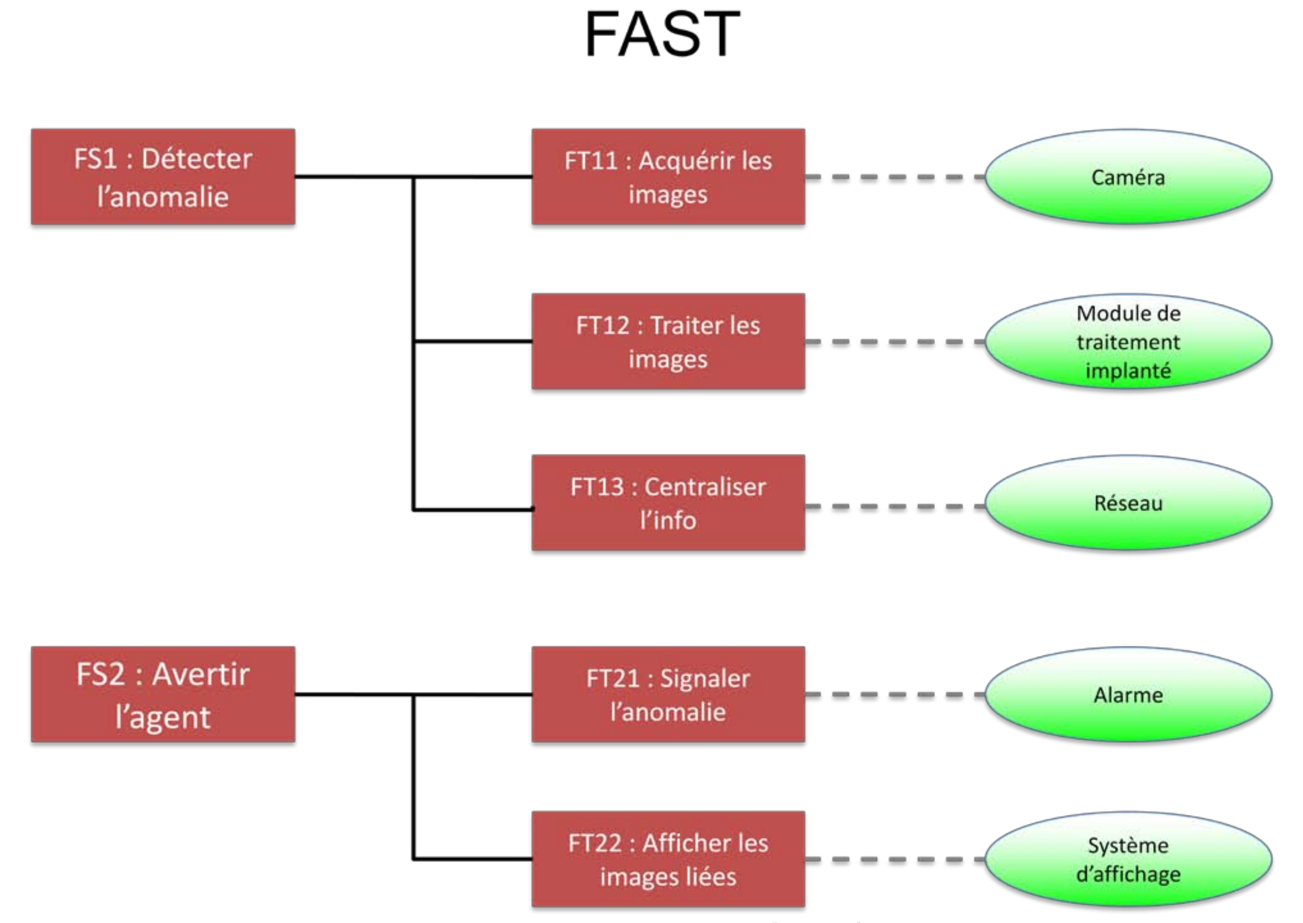


Figure 2 : Diagramme FAST du système.

Légende :
FS : Fonction de service
FT : Fonction technique

Détail des fonctions techniques – Choix des solutions techniques

FS1 : Détecter l'anomalie

FT11 : Acquérir les images
• On a besoin d'une fréquence d'acquisition supérieure à 5 Hz, mais pas nécessairement d'atteindre une valeur très élevée.
• On a besoin d'une certaine résolution d'image : le standard VGA (640x480) suffit.
• On n'a pas besoin d'une image en couleur.

FT12 : Traiter les images
• On a besoin d'un système informatique et d'outils programmables de traitement du signal. Un logiciel de traitement comme MATLAB pourrait suffire, mais transmettre sans perte les données venant de 25 caméras rendrait la fonction FT13 irréalisable par le réseau informatique de l'école, qui est la seule solution technique utilisable pour nous.
• On doit donc réaliser le traitement d'image avant l'étape de transmission (c'est pourquoi le traitement est en FT12 et pas FT13). On utilise donc un module de traitement implanté directement au niveau de la caméra.

FT13 : Centraliser l'information
• On doit pouvoir véhiculer des informations, mais nous ne sommes pas en mesure de mettre en place un réseau de câbles à cet effet. On lui préférera le réseau Internet de l'école.
• On doit donc pouvoir repérer les caméras par adresse IP.

On décide donc de choisir, pour remplir la fonction de service FS1, une caméra repérable par IP, équipée d'un module de traitement d'image interne et de connectique de mise en réseau (Ethernet), capable d'acquérir un flux vidéo noir et blanc VGA : le modèle retenu compte tenu des prix et de l'utilisation faite pour réaliser un prototype sera « BlueLYNX 400 GX » du constructeur MatrixVision.

FS2 : Avertir l'agent

FT21 : Signaler l'anomalie
• L'alerte doit être au moins sonore, c'est la seule restriction. On peut donc utiliser n'importe quel système d'alerte.

FT22 : Afficher les images
• On doit pouvoir amener les images non compressées par réseau dans le local de sécurité. Il faut y trouver un ordinateur relié au réseau de l'école, ce qui existe peut-être déjà, mais n'est pas difficile à mettre en place.

La réalisation de la Fonction de Service 2 n'impliquera que des modifications superficielles de l'installation du local de sécurité. Il faudra consulter les agents.

Ces deux outils de gestion de projet nous ont permis de diviser le projet en plusieurs phases. Ce travail représente le socle nécessaire à l'élaboration du GANTT.

II.2- *Elaboration du GANTT*

Le diagramme GANTT est un outil de gestion de projet qui permet de visualiser dans le temps les différentes tâches du projet. Cela permet de planifier le projet et par conséquent, de donner une dynamique au projet. Pour représenter ce GANTT, nous avons utilisé le logiciel GANTT Project.

1ère phase : début Mai jusqu'à fin Juin 2009

La figure 3 (gauche) représente le GANTT correspondant à la première phase du projet.

2ème phase : début Septembre 2009 jusqu'à mi-Avril 2010-04-16

Malheureusement, certains problèmes sont apparus pendant le projet, ce qui nous a retardé. En effet, nous n'avons pas réussi à faire fonctionner la caméra que nous avions achetée. C'est pourquoi nous avons établi un nouveau diagramme GANTT qui correspond mieux à l'état réel d'avancement du projet ainsi que de la durée de travail que nous avons accordée à chacune des tâches.

La figure 3 (droite) représente le GANTT correspondant à cette deuxième phase du projet.

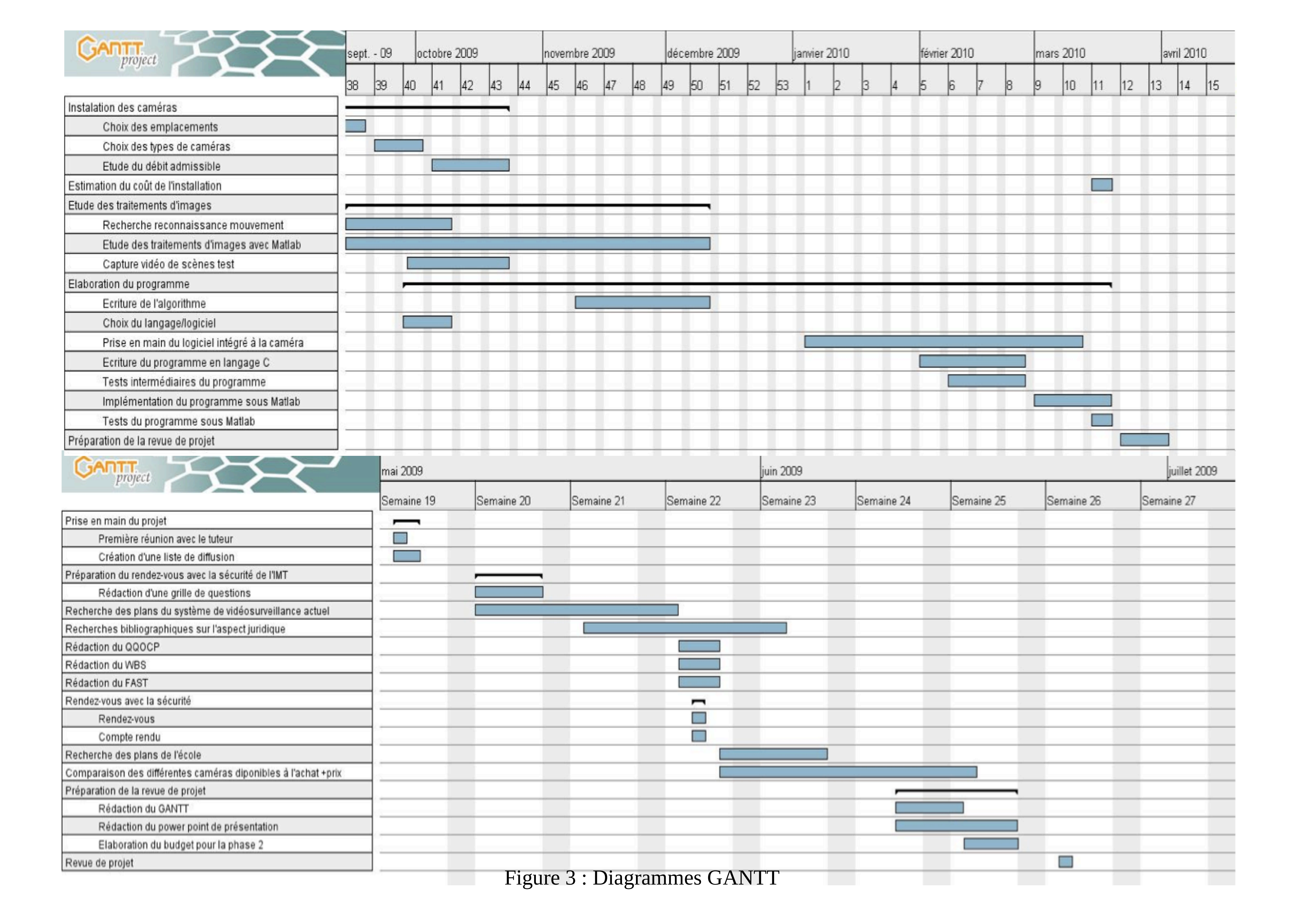


Figure 3 : Diagrammes GANTT



II.3- *Organisation de l'équipe projet*

Pour répartir les membres du projet sur les différentes tâches du projet, nous avons utilisé la fonction « ressources humaines » du logiciel GANTT Project. Voici l'affectation des tâches à chaque membre du groupe d'après le GANTT précédent.

Sébastien Angot :
- Choix des types de caméras
- Estimation du coût de l'installation
- Capture vidéo de scènes tests
- Prise en main du logiciel intégré à la caméra

Godefroy Brisebarre :
- Choix des types de caméras
- Etude du débit admissible
- Capture vidéo de scènes tests
- Ecriture de l'algorithme
- Ecriture du programme en langage C

Charlotte Cadoret :
- Choix des emplacements
- Etude du débit admissible
- Capture vidéo de scènes tests
- Ecriture de l'algorithme
- Tests intermédiaires du programme

Marc Navatier :
- Etude des traitements d'images avec MATLAB
- Prise en main du logiciel intégré à la caméra

David Nicolet-Dit-Félix :
- Choix des emplacements
- Etude des traitements d'images avec MATLAB
- Ecriture de l'algorithme
- Choix du langage/logiciel
- Ecriture du programme en langage C
- Tests du programme sous MATLAB

Damien Pineau :
- Etude des traitements d'images avec MATLAB
- Ecriture du l'algorithme
- Choix du langage/logiciel
- Ecriture du programme en langage C
- Implémentation du programme sous MATLAB
- Tests du programme sous MATLAB

II.4- *Fonctionnement durant le projet*

Pour faire fonctionner la caméra, nous avions besoin d'un ordinateur sous Linux. L'école nous a prêté un ordinateur portable pour la durée du projet à l'usage des membres du groupe qui travaillaient à prendre en main la caméra MV Blue Lynx.

Pour faire le point sur le projet, nous avions rendez-vous chaque semaine avec notre tuteur, M. Ali Khalighi. Cela nous permettait de rendre compte de l'avancement de chacune des tâches et d'identifier rapidement les problèmes auxquels pouvait être confronté chaque membre.

D'autre part, nous avons eu des réunions régulières avec le client destinées à nous orienter selon les besoins du projet.

II.5- *Budget*

L'école a alloué 500 euros à chaque groupe de projet transverse pour des achats nécessaires à l'avancement du projet.

Notre projet étant basé sur l'étude et le développement d'un réseau de vidéosurveillance « intelligente », les dépenses nécessaires se limitent à l'achat d'une caméra « intelligente », elle aussi, comprenant un système de traitement embarqué, capable d'être l'hôte d'un programme de détection développé par nos soins.

Dans ce cadre, nous avons pris contact avec la société MatrixVision qui nous a proposé des caméras dont le premier prix s'élève à 1190 euros HT, ce qui reste plus abordable que ce que proposent d'autres entreprises.

Nous avons donc négocié avec l'entreprise pour obtenir une solution dans nos moyens. Ainsi après un entretien téléphonique et quelques échanges écrits, notre contact nous a proposé un exemplaire d'occasion, ayant servi pour des démonstrations client et en parfait état de marche pour la somme de 500 euros HT, ce qui rentre bien plus dans les contraintes du budget. Des discussions ont eues lieu en ce qui concerne les accessoires fournis avec la caméra, un starter-kit étant vendu 100 euros HT (au lieu de 890) avec un exemplaire neuf de la caméra, nous aurait convenu. Mais cette réduction ne peut s'appliquer à l'achat d'un exemplaire d'occasion. Nous avons réussi à faire descendre le prix à 150 euros HT au lieu de 890. Cela nous fait un budget total de 650 euros HT, certes plus élevé que les crédits alloués, mais le manque à perdre par rapport au prix d'un matériel neuf est plus qu'avéré.

D’autre part, voici une estimation du budget total nécessaire à l’installation d’un système de vidéosurveillance opérationnel utilisant des caméras « intelligentes » comprenant un système de traitement embarqué.

Les prix fournis pour les caméras ainsi que leurs accessoires par la société MatrixVision sont les suivants :

Désignation	**PUHT**	**Quantité**	**PTHT**	**TVA 19,6%**	**PTTTC**
mvBlueLYNX-400GX Caméra CMOS 640x480, cpu 200Mhz, 32MB RAM, 32MB Flash, out VGA/video	1 190 €	25	29750 €	5831 €	35581 €
Objectif monture C 16mm F1.4, capteur jusqu’au 2/3 pouce, MOD 0,4m, vis de blocage	95 €	25	2375 €	465,50 €	2840,50 €
MV-DC2425 BL Alimentation 24 Volts pour BlueLYNX type 2XX et supérieur	175 €	25	4375 €	857,50 €	5232,50 €
Totaux			36500 €	7154 €	43654 €

Figure 4 : Prix total de 25 caméras

Le projet ne comportant pas de réalisation concrète dudit réseau, nous ne sommes pas en mesure de fournir certains devis, car il faudrait pour cela réaliser un appel d’offre, notamment dans le cas d’installation de câbles et prises réseau et d’alimentation.

Nous devons donc ajouter les coûts d’installation de nouvelles prises réseau proches des caméras, que nous estimons, suite à une analyse de différents devis fournis par le CRI, de l’ordre de la dizaine de milliers d’euros toutes taxes comprises.

Nous aurons également besoin de caches de protection pour les caméras extérieures, mais également de prises d’alimentation et de supports de fixation pour les nouveaux emplacements de caméras.

Nous pouvons donc estimer le coût total de l’installation aux alentours des 60 000 euros. Ce budget pourrait néanmoins être revu à la baisse, en négociant les prix au vu de la quantité de produits et services demandés, ainsi qu’en faisant jouer la concurrence.

III- Etude préliminaire de l'organisation globale de la vidéosurveillance

III.1- *Plan de localisation des caméras*

Pour choisir la localisation des caméras sur le site de l'IMT, nous avons pris rendez-vous avec les agents de sécurité du site pour comprendre quels sont les réels besoins en matière de vidéosurveillance. En effet, il s'avère que des caméras supplémentaires, notamment près du foyer des élèves seraient utiles. Nous avons donc élaboré un plan de localisation des caméras dans l'optique du nouveau système de vidéosurveillance.

Plan de l'IMT :

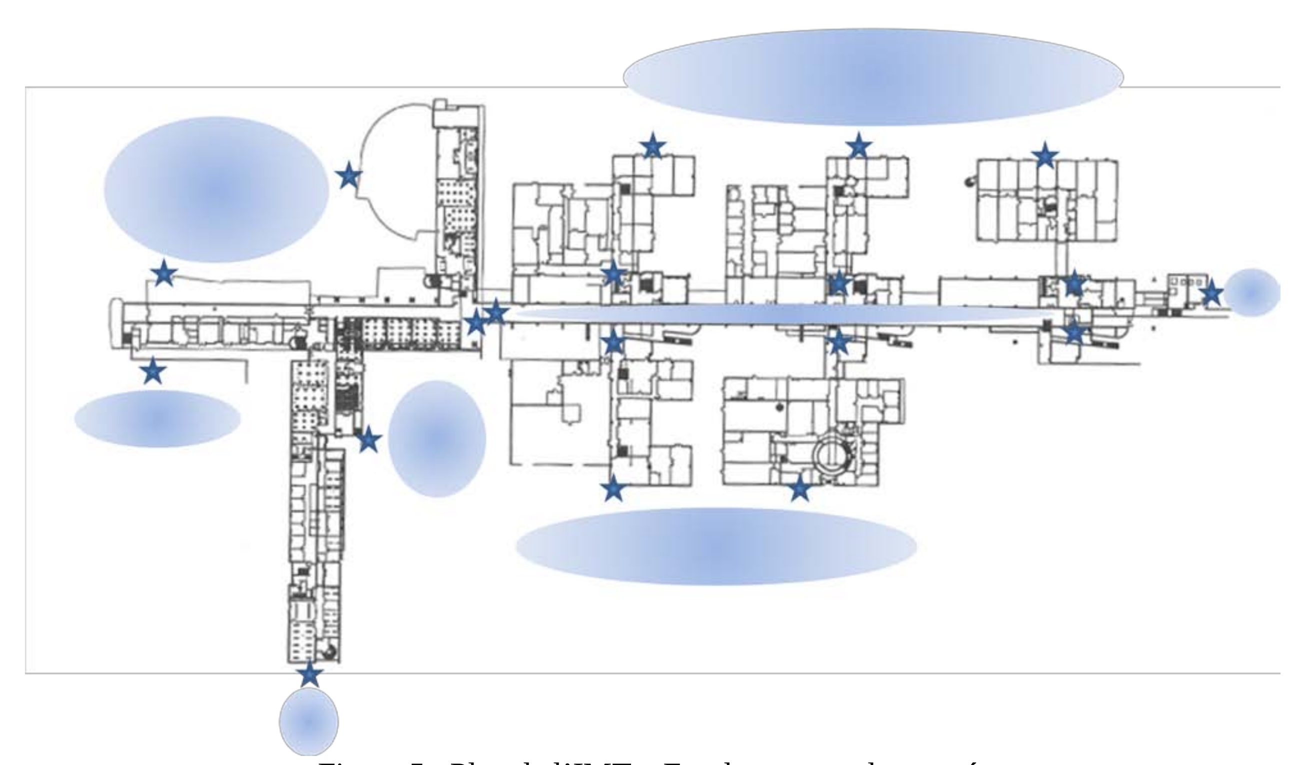


Figure 5 : Plan de l'IMT – Emplacements des caméras

Les étoiles correspondent aux caméras que l'on souhaite mettre en place. On visualise également les champs des différentes caméras.

III.2- *Aspects juridiques de la vidéosurveillance*

Le cadre juridique de la vidéosurveillance est très précis de par le fait que la liberté des personnes filmées est engagée. La surveillance des voies publiques est strictement interdite, de même que la surveillance des zones privées d'autrui. Il est donc nécessaire d'adapter les champs des caméras de façon à ce qu'aucune de ces zones ne soit dans le champ.

Se pose ensuite le problème de l'enregistrement des images recueillies. Si on choisit de ne pas conserver les données recueillies, conformément aux statuts de l'Ecole Centrale Marseille, aucune autorisation autre n'est demandée. Au contraire, si on souhaite enregistrer les vidéos, une autorisation de la CNIL est nécessaire. De plus, si on veut utiliser les vidéos à des fins juridiques, des normes techniques supplémentaires sur la résolution et la fréquence des images sont demandées.

III.3- *Deux solutions possibles : le traitement centralisé ou le traitement embarqué des images*

Suite à notre étude préliminaire du problème et à notre travail de veille technologique, nous avons dégagé deux solutions qui nous permettent de réaliser les exigences du client : le traitement centralisé des images ou le traitement embarqué. Ces deux solutions se situent à des niveaux budgétaires très différents puisque la dernière nécessite l'acquisition de caméras très particulières et relativement coûteuses alors que l'autre se contente de caméras de surveillance standard.

Le traitement embarqué des images présente néanmoins certains avantages supplémentaires, notamment en ce qui concerne la facilité de mise en place du système et, plus particulièrement, la mise en place du réseau qui véhicule les flux audiovisuels. En effet, elle est très simple à mettre en place en partant du réseau informatique actuel de l'école et elle ne nécessite que quelques modifications. Le traitement centralisé des images, quant à lui, nécessite une installation plus complexe, à savoir l'installation de nombreux câbles sur l'ensemble du site de l'école. Cette installation étant rendue d'autant plus difficile que, le site de l'IMT n'étant pas la propriété exclusive de l'école, il est nécessaire pour effectuer les travaux d'installation d'obtenir l'aval de nombreux responsables.

Sur le plan technique, la différence majeure entre les deux solutions est la localisation physique du système de traitement des images et de détection de mouvement. C'est cette différence qui engendre les variations de coût et de facilité d'installation entre les deux solutions envisagées. En effet, soit le système reçoit toutes les images et les traite au poste de sécurité – ce qui pourrait poser des problèmes d'engorgement de réseau si ce dernier est utilisé pour transporter les

informations – soit il effectue le traitement au niveau de chaque caméra, qui embarque alors un processeur nécessaire pour effectuer le traitement des images à la source, et ainsi n'envoie la totalité des informations recueillies par la caméra qu'en cas d'anomalie, pour une utilisation plus économe quant au débit d'information disponible sur un réseau.

Une autre différence provient du fait que la réalisation du traitement des images sur les caméras implique de ne pas utiliser des programmes qu'on ne peut pas implanter sur celles-ci, ce qui a pour conséquence essentielle que notre programme de traitement doit être écrit en langage C ou C++ et nécessiter peu, voire pas, d'appels à des programmes de calcul tels que MATLAB, SCILAB ou LABVIEW.

D'un point de vue législatif, le traitement des images à la source permet d'éliminer plus simplement et plus efficacement les images prises par les caméras de zones que l'on ne peut filmer puisque cette application d'un masque est propre à chaque caméra.

Nous avons présenté ces deux solutions à notre client et à notre tuteur école. Leurs conseils nous ont orientés vers le choix de la solution où le traitement des images est réalisé sur la caméra. Nous allons par la suite développer plus précisément cette solution retenue.

Enfin, les différentes solutions envisagées ne différaient pas exclusivement sur ces points mais aussi sur d'autres tels que le fait de filmer en couleur ou non, le fait de filmer en infrarouge la nuit ou d'éclairer la scène. Cependant ces autres points n'engendrent pas de différences considérables dans la solution finale, c'est pourquoi on ne s'attardera pas dessus.

IV- Présentation de la solution retenue et prise en main du matériel

IV.1- *Solution utilisant le système de traitement embarqué*

Bien que cette solution soit la plus coûteuse, nous avons choisi celle-ci car elle nous permet de traiter plus efficacement les images, notamment puisqu'on évite l'apparition de bruit dû au transport de l'information. Le point qui est majoritairement à l'origine de cette décision reste la simplicité de mise en œuvre du système sur le site de l'IMT.

En ce qui concerne les différents autres critères, nous avons opté, au vu des caractéristiques techniques des caméras qui embarquent un microprocesseur, pour les solutions permettant de réduire le coût du projet. Ainsi, nous avons choisi de filmer en noir et blanc et d'éclairer les scènes nocturnes plutôt que de considérer des caméras qui filmeraient aussi en infrarouge.

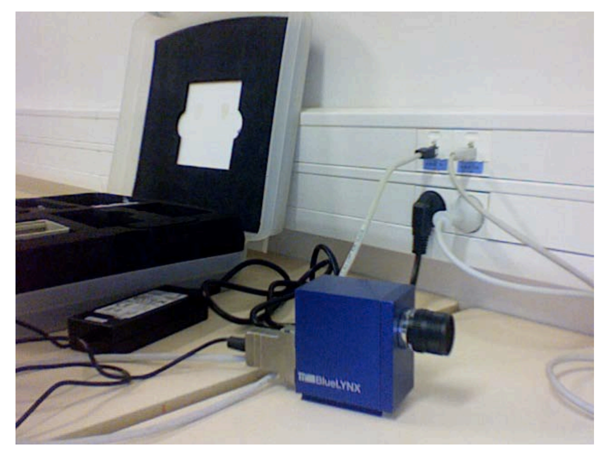
De façon plus précise les caméras retenues sont les modèles de la gamme BlueLynx réalisées par MatrixVision, une société allemande possédant une filière française.

Ces caméras présentent l'avantage d'intégrer de base le nécessaire afin d'implanter un programme en C ou C++ dessus permettant le traitement des images.

Après consultation avec le Centre des Ressources Informatiques de l'école, il nous est apparu que la meilleure solution pour réaliser le réseau qui relierait ces caméras au centre de sécurité est de créer un sous-réseau virtuel sécurisé et totalement indépendant du reste du réseau de l'école, bien qu'il utilise les mêmes installations.

IV.2- *Prise en main de la caméra MV-BlueLynx (MatrixVision)*

La caméra se présente comme un bloc disposant, pour ce qui est des protocoles de communication, d'un port série et d'un port Ethernet.



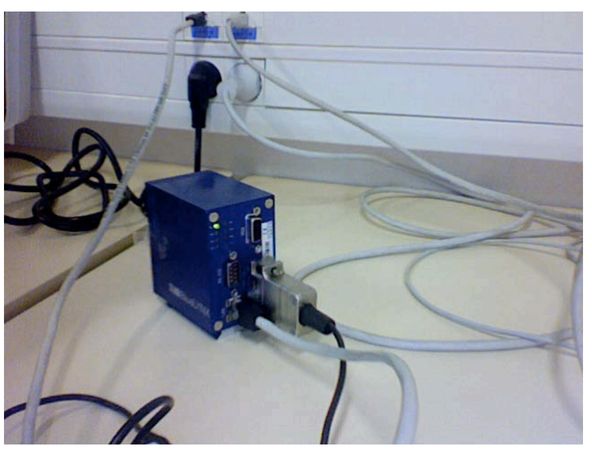


Figure 6 : La caméra MV BlueLynx, installée dans une salle de l'Equerre

Dans un premier temps, il nous a fallu utiliser le protocole filaire série pour initialiser la caméra et lui attribuer une adresse IP fixe (10.228.0.29) dans le réseau local des ports Ethernet pour ordinateurs portables de l'Equerre.

Ensuite, via une connexion internet et l'utilitaire de lignes de commande UNIX Telnet, nous avons pu communiquer avec la caméra et commencer à explorer les dossiers qu'elle contient.

```
Terminal — telnet — 80×24
MacBook-de-Marc-Navatier:~ marcnavatier$
MacBook-de-Marc-Navatier:~ marcnavatier$
MacBook-de-Marc-Navatier:~ marcnavatier$
MacBook-de-Marc-Navatier:~ marcnavatier$
MacBook-de-Marc-Navatier:~ marcnavatier$
MacBook-de-Marc-Navatier:~ marcnavatier$ telnet 10.228.0.29
Trying 10.228.0.29...
Connected to 10.228.0.29.
Escape character is '^]'.

MATRIX Vision mvBlueLYNX (mvs) Linux ppc (2.4.20)
set alias on mvBlueLYNX
mvs:/#
```

Figure 7 : Confirmation de connexion à la caméra IP

Nous y avons trouvé certains programmes de test déjà compilés (« mvstest » …) qui ont pour but de montrer comment la caméra peut envoyer des informations sur internet, en les rendant disponibles sur le port 10 000 de son adresse IP (protocole http).

```
Terminal — telnet — 80×24
mvs:/# ls flash/mvstest
hist.lst      image4.pgm    image7.pgm    imageserver   mvstest
image0.pgm    image5.pgm    image8.pgm    mvserver
image3.pgm    image6.pgm    image9.pgm    mvserver_new
mvs:/# ls
bin        etc        lib        old_root  sbin       var
data       flash      linuxrc    proc      tmp
dev        home       mnt        root      usr
mvs:/#
mvs:/# ls
bin        etc        lib        old_root  sbin       var
data       flash      linuxrc    proc      tmp
dev        home       mnt        root      usr
mvs:/#
mvs:/#
mvs:/# ls flash
home      lib       mvstest  picogui   tools
mvs:/#
mvs:/#
mvs:/# ls flash/mvstest
hist.lst      image4.pgm    image7.pgm    imageserver   mvstest
image0.pgm    image5.pgm    image8.pgm    mvserver
image3.pgm    image6.pgm    image9.pgm    mvserver_new
mvs:/#
```

Figure 8 : Le répertoire contenant les programmes test, flash/mvstest

Par exemple, le programme mvserv envoie une image noire en jpeg (64x480) à l'url http://10.228.0.29:10000/

```
Terminal — telnet — 80×24
image3.pgm    image6.pgm    image9.pgm    mvserver_new
mvs:/# ls
bin        etc        lib        old_root  sbin       var
data       flash      linuxrc    proc      tmp
dev        home       mnt        root      usr
mvs:/#
mvs:/# ls
bin        etc        lib        old_root  sbin       var
data       flash      linuxrc    proc      tmp
dev        home       mnt        root      usr
mvs:/#
mvs:/#
mvs:/# ls flash
home      lib       mvstest  picogui   tools
mvs:/#
mvs:/#
mvs:/# ls flash/mvstest
hist.lst      image4.pgm    image7.pgm    imageserver   mvstest
image0.pgm    image5.pgm    image8.pgm    mvserver
image3.pgm    image6.pgm    image9.pgm    mvserver_new
mvs:/# cd flash/mvstest
mvs:/flash/mvstest# mvserver_new
Listening at port 10000
```

Figure 9 : Exécution du programme test mvserver_new



Nous avons également repéré dans l'arborescence le répertoire intitulé « ftp » qui correspond bien sûr au serveur ftp intégré à la caméra.

La caméra contient un certain nombre de librairies de programmation, dont le détail des fonctions a été difficile à obtenir, du fait d'un manuel d'utilisation très peu clair et de difficultés de communication avec les contacts que nous avons au sein de MatrixVision, le constructeur du modèle.

```
Terminal — telnet — 80×24
mvs:/flash/mvstest# cd ..
mvs:/flash# ls
home      lib      mvstest  picogui  tools
mvs:/flash#
mvs:/flash#
mvs:/flash#
mvs:/flash# ls lib
ipkg                       libvl_autotrace.so.5.1.1  libvl_ocr.so
libTreatments.so           libvl_base.so             libvl_ocr.so.5
libmvblacquire.so          libvl_base.so.5           libvl_ocr.so.5.1.1
libmvblacquire.so.3        libvl_base.so.5.1.1       libvl_slice.so
libmvblacquire.so.3.4.5    libvl_device.so           libvl_slice.so.5
libpgserver.so             libvl_device.so.5         libvl_slice.so.5.1.1
libpgserver.so.1           libvl_device.so.5.2.7     libvooi_aoi.so
libpgserver.so.1.0.5       libvl_display.so          libvooi_aoi.so.5
libpng.so                  libvl_display.so.5        libvooi_aoi.so.5.1.1
libpng.so.2                libvl_display.so.5.2.5    libvooi_loi.so
libpng.so.2.1.0.12         libvl_file.so             libvooi_loi.so.5
libthread_db-1.0.so        libvl_file.so.5           libvooi_loi.so.5.1.1
libthread_db.so            libvl_file.so.5.1.1       libvooi_poi.so
libthread_db.so.1          libvl_ipl.so              libvooi_poi.so.5
libvl_autotrace.so         libvl_ipl.so.5            libvooi_poi.so.5.1.1
libvl_autotrace.so.5       libvl_ipl.so.5.1.1
mvs:/flash#
```

Figure 10 : Certaines des librairies que contient la caméra.

C'est ici que nous nous heurtons à un mur apparemment infranchissable : la documentation donne des explications bien trop sporadiques et incompréhensibles sur l'utilisation des librairies et des fonctions qu'elles contiennent dans le cadre de la rédaction d'un programme, ce qui est frustrant, car on a pu concevoir par ailleurs un programme traitant en direct les images provenant d'une webcam.

On a d'un côté les outils et de l'autre côté l'idée, mais on n'arrive pas à l'appliquer à notre cas précis avec l'aide de ces outils, car on ne sait pas comment s'en servir. Les fonctions de traitement intégrées à l'appareil sont nombreuses et précises, elles suffiraient à réaliser en langage C un programme équivalent à celui que l'on a obtenu sous MATLAB, bien que cela semble très fastidieux.



V- Réalisation du programme de traitement d'images

V.1- *Algorithme retenu*

Les caméras génèrent un flux vidéo en noir et blanc. Pour traiter le flux vidéo, l'algorithme devra comporter les points suivants :

- séquencer le flux vidéo en images
- comparer les images à une image de référence
- détecter les éventuelles anomalies
- signaler/mettre en évidence ces anomalies

Pour écrire ce programme, nous avons dû prendre des images tests (certaines d'internet, d'autres de vidéos que nous avons tournées, d'autres d'une webcam). Nous avons écrit un programme en C puis un en MATLAB.

V.2- *Programmation en langage C*

On considère une image de référence et une image dite "test" différente (a priori) de l'image de référence.

La première phase du programme consiste en une simple différence entre les deux images, alors considérées comme des matrices de dimensions 640x480 (résolution de l'image de la caméra) dont chacune des valeurs correspond à un niveau de gris (moyenne des trois valeurs RGB, le 0 correspond au noir et le 255 au blanc). En prenant la valeur absolue, on obtient alors une matrice à valeurs comprises entre 0 et 255.

La deuxième phase du programme consiste en un seuillage de cette matrice de différence. On fixe un seuil. Toutes les valeurs au-dessous de ce seuil passent à 0 et toutes celles au-dessus passent à 255. On obtient alors une matrice avec comme valeurs soit 0 soit 255, c'est une image binaire en noir et blanc. Dans cette image, on a encore des pixels bruités qui apparaissent (en blanc).

La troisième étape du programme consiste en une érosion de l'image obtenue par seuillage. Pour ce, on applique un filtre sur toute la matrice. Un filtre est un ensemble de pixels. À chaque endroit où le filtre passe, on regarde la valeur de chacun des pixels du filtre. Si tous les pixels du filtre sont blancs, le pixel-clé (celui du centre dans notre cas) devient blanc sur l'image érodée. Dans le cas contraire, ce pixel devient noir si au moins un pixel du filtre est noir. Nous avons essayé plusieurs filtres : carré 3x3, croix 3x3, carré 5x5. Nous avons retenue la solution du carré 5x5. En effet, le carré 3x3 n'élimine pas forcément toutes les imperfections et la croix n'a pas de grand intérêt. Ainsi on a toujours une image binaire où les pixels bruités ont disparu. Cependant, les surfaces ont rétréci.



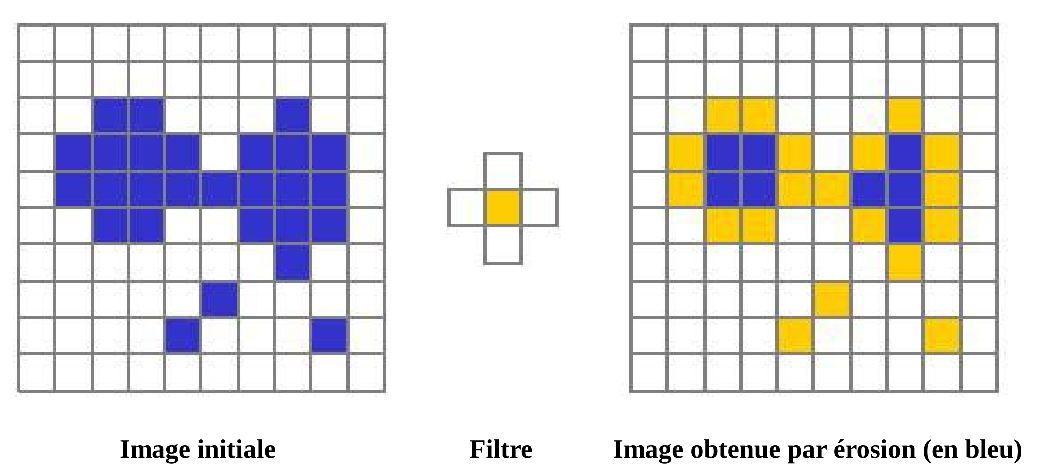


Figure 11 : Traitement d’image - érosion

La quatrième étape consiste en une dilatation des surfaces précédemment érodées. Pour ce, on applique un autre type de filtre. Concrètement, on considère un ensemble de pixels (le même que celui du filtre) et quand au moins un est blanc, le pixel-clé (du centre) devient blanc sur l'image obtenue par dilatation. Ainsi, on obtient les mêmes surfaces principales que l'image après seuillage mais sans les imperfections. On obtient au final une image noire et blanche où les pixels blancs traduisent un événement inhabituel.

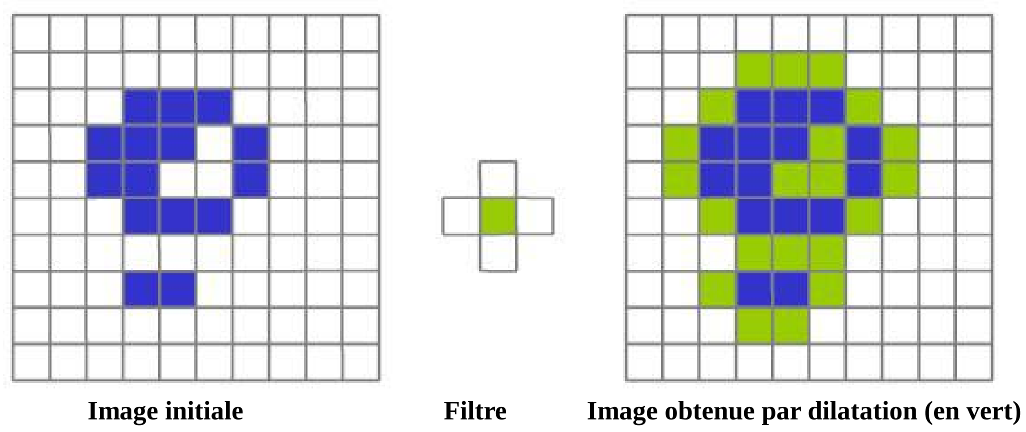


Figure 12 : Traitement d’image - dilatation

L'action d'éroder puis de dilater une image, comme nous l'avons fait, s'appelle l’ouverture d’une image.

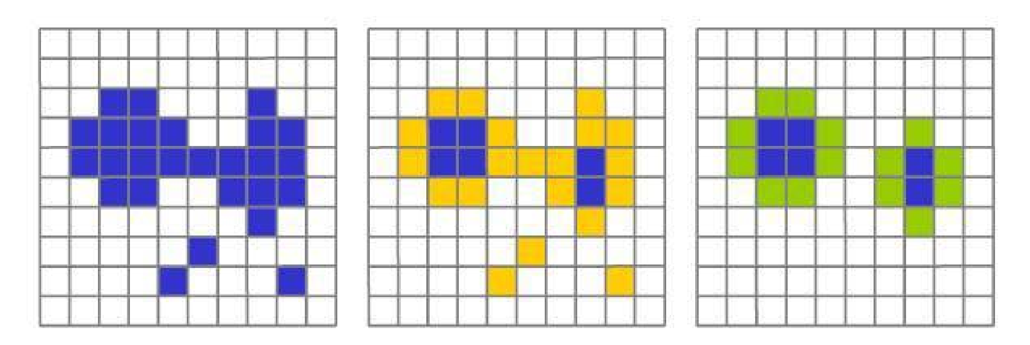


Figure 13 : Traitement d’image – érosion puis dilatation



V.3- *Prototypage sous MATLAB*

Sous MATLAB, on crée le programme sous forme de schéma bloc avec des fonctions de transfert à paramètres réglables à l'aide de la toolbox « Video and image processing ». Le prototype réalisé est situé en annexe.

Pour séquencer les images, il suffit de régler le paramètre de la fonction qui séquence les images sur la valeur souhaitée. Après plusieurs tests, nous avons estimé que 1 image sur 5 était suffisant pour garder une certaine fluidité, ce qui laisse au programme un peu moins de 0,2 sec pour traiter et afficher l'image.

L'image sélectionnée est comparée à la précédente, non traitée, et mise en mémoire. Pour comparer les images, on ne raisonne pas par pixel mais par zone. Concrètement on sépare l'image en sous-images (ou « blocs »), par exemple, l'image fait 640x480 pixels, on la divise en m x n images de (640/m)x(480/n) pixels ; m et n étant des paramètres réglables. Pour comparer les images, on calcule la valeur absolue de la différence entre la valeur d'un pixel de l'image et celle du même pixel de l'image précédente, puis on somme les différentes valeurs obtenues dans chaque bloc. Cette opération est effectuée dans le Block 'processing 1' en annexe.

On compare ensuite cette valeur à un niveau fixé manuellement : si la valeur est au dessus de ce niveau, on affiche la sous-image de la manière expliquée ci-après, dans le cas contraire, on ne fait rien. Ce seuil doit être déterminé à chaque nouvel emplacement en lançant le programme lorsqu'il n'y a pas de mouvement.

Dans ce prototypage, l'ouverture de l'image n'est pas utilisée car elle n'est pas nécessaire. En effet, les pixels éliminés par l'érosion n'ont maintenant plus d'importance si on choisit un seuil adapté. De plus, en n'utilisant pas ces outils, le temps de calcul du traitement s'en trouve diminué.

Pour mettre en évidence les sous-images sujettes à un mouvement, on affiche en sortie une image en couleur en RGB. Les sous-images sans mouvement notable restent en noir et blanc et les autres ont pour valeur sur les canaux G et B la valeur du niveau de gris et deviennent sur le canal R le complémentaire du bloc. Ainsi les parties blanches de la sous-image apparaitront en bleu et les parties noires en rouge sur l'image finale. Cette action débute dans le Block 'processing 2' ou 'Highlight' qui se trouve en annexe.

Ainsi en affinant assez la division en sous-images, on peut obtenir une zone précise où le mouvement a eu lieu, cependant le temps de calcul augmente aussi.

Une autre fonction située au début du prototype permet de ne pas traiter une zone de l'image afin de respecter certains aspects juridiques ou d'éliminer des zones où le traitement n'est pas nécessaire, comme le mouvement d'arbres. Pour ce faire, on applique un masque en multipliant le flux vidéo reçu par une matrice binaire (composée de 0 ou de 1). Ainsi les zones à ne pas traiter deviendront noires et donc aucune anomalie ne sera détectée. Ce masque dépend évidemment de l'emplacement de la caméra.

V.4- *Tests et présentation des résultats*

- Pour le programme en C, nous avons pris une image de référence,



Figure 14 : Image de référence

puis nous avons pris une autre image de la vidéo avec un « intrus » dessus.



Figure 15 : Image à traiter

On effectue la différence et on effectue un seuillage pour obtenir une image en noir et blanc :



Figure 16 : Image après seuillage

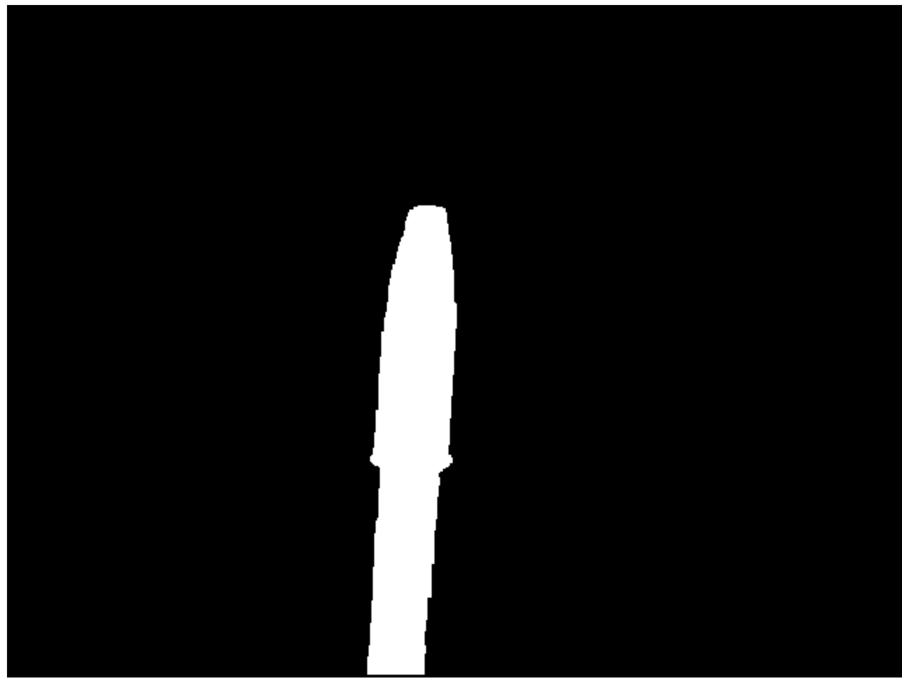
Après l'érosion :



Figure 17 : Image après seuillage et érosion



Après la dilatation :


Figure 18 : Image après seuillage, érosion et dilatation



- Prototype sous MATLAB

Toutes ces images ne sont qu'une représentation de la vidéo en sortie mise sous forme de séquence d'image. Les zones en couleur indiquent les endroits où il y a eu détection d'un mouvement. Pour ce faire, chaque image a été segmentée en 64 blocs.

Images prises toutes les 2 secondes.

Figure 19 : Détection et mise en valeur de mouvement

Pour chaque bloc, on obtient une valeur comme décrit dans le prototypage sous MATLAB. Ces valeurs permettent de déterminer le niveau de référence à partir duquel on détecte un mouvement significatif. Au cours du temps, ces valeurs forment des courbes qui permettent de distinguer les mouvements. Pour l'exemple, le premier graphique ci-dessous contient seulement 5 courbes :

- le niveau de référence en vert
- 4 courbes représentant les valeurs des 4 blocs et donc une zone

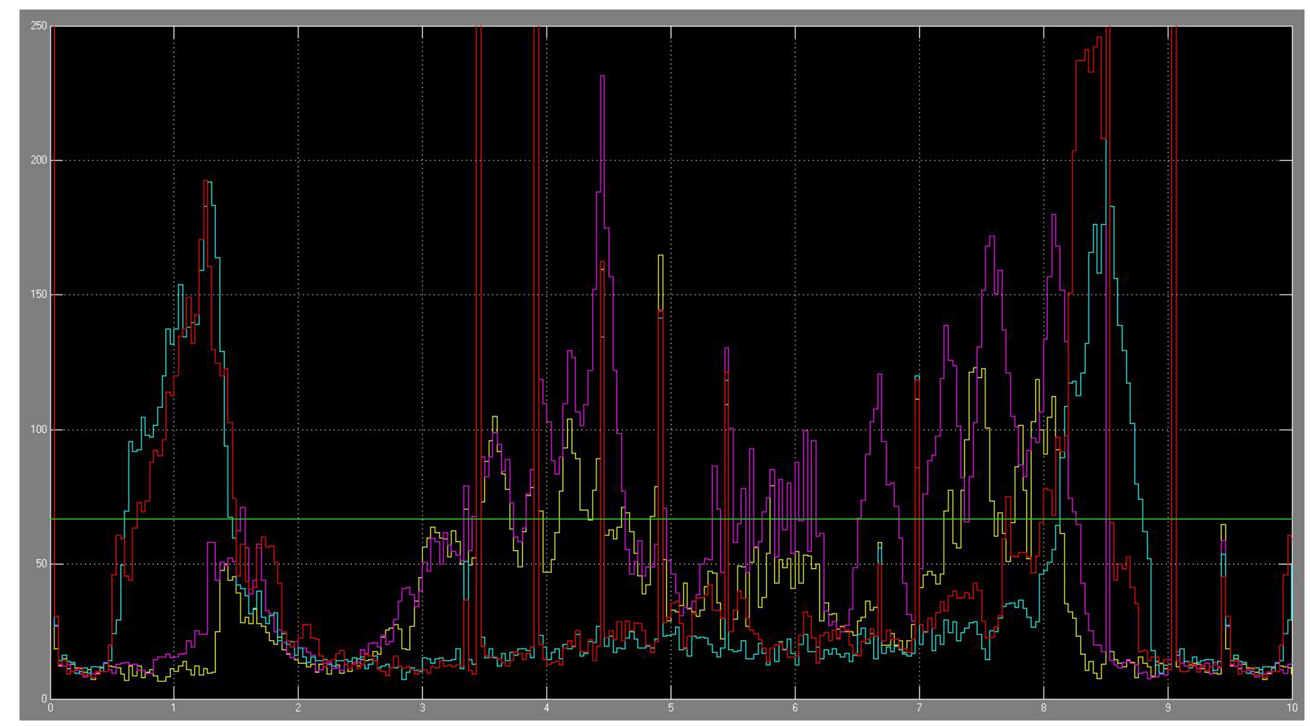


Figure 20 : Courbes de différences entre images (bloc par bloc)

Si la courbe est au dessus du niveau alors il y a détection de mouvement dans une zone. Pour fixer, on effectue un test en filmant une zone où il n’y a pas de mouvement comme la zone centrale de la prochaine image qui compte 65 courbes donc pour un traitement avec 64 blocs.

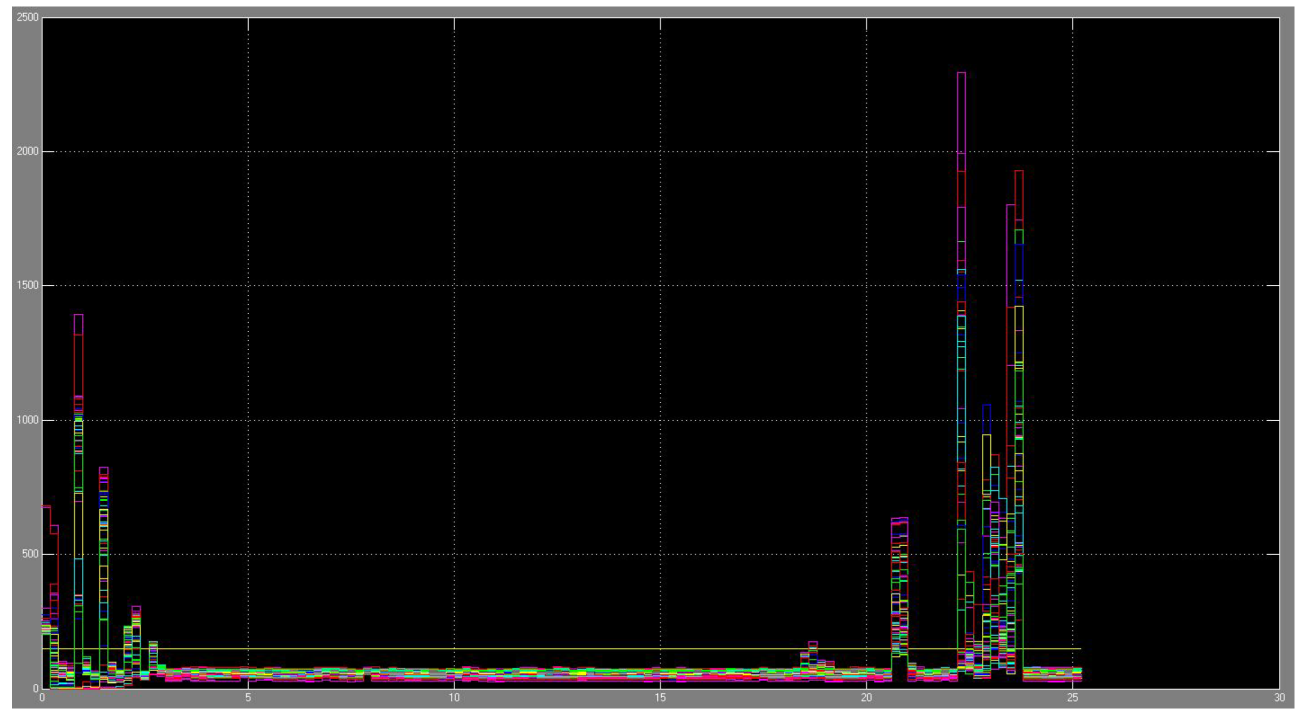


Figure 21 : Réglage du seuil de détection

On remarque 3 zones, la dernière étant seulement un essai pour voir si le seuil est adapté.

La première zone du graphique est due à la webcam qui effectue un réglage automatique afin d’améliorer le contraste de l’image.

La deuxième zone est celle qui permet d’évaluer le niveau de détection. Les courbes ne sont pas nulles à cause du bruit dû à l’acquisition par la webcam du flux vidéo. Il convient donc de choisir un niveau supérieur à ce bruit. Cependant il faut tenir compte d’autres facteurs. Le mouvement d’arbres peut être éliminé en choisissant une limite plus importante mais cela peut nuire à la détection d’autres anomalies. Cette limite, tout comme le masque, est une solution envisageable pour supprimer les détections qui ne sont pas significatives.

V.5- *Limites de notre programme*

Notre programme n’est qu’une ébauche, présentant des écarts vis-à-vis des objectifs sous de nombreux aspects.

Tout d’abord, on a de grandes limites de compatibilité. Comme cela a été dit précédemment, notre concept de traitement de la vidéo pour détecter des anomalies ne se matérialise pour l’instant qu’à travers un programme réalisé sous MATLAB, logiciel lourd et licencié, mais surtout vraisemblablement non-installable sur une caméra à module informatique telle que les modèles BlueLynx de MatrixVision.

Une autre limite est la courante latitude de mise au point, c’est-à-dire l’équilibre acceptable entre la sensibilité et l’intelligence du système de détection. On a vu que la détection commence par une étape de seuillage, dont le seuil est fixé par l’humain. En effet, il existe des variations pouvant survenir entre deux images prises par une caméra qu’il n’est pas souhaitable de signaler (cas de la pluie, du vent dans les feuillages d’arbres, etc.). Il est bon de régler ce seuil de façon à négliger ces variations, mais il ne faut pas que ce soit en dépit de la détection de réelles anomalies.

On rencontre aussi une limite de rapidité de notre programme. Le calcul se fait sur un certain nombre de blocs qui fractionnent l’image. L’idéal est de travailler en segmentant l’image en un nombre important de blocs, pour affiner la détection spatiale. Cependant, passé un certain nombre de blocs, le calcul est trop conséquent et le traitement en temps réel n’est plus possible.

Enfin, tel qu’il est conçu, notre programme ne peut pas être mis en place sur une caméra mouvante (il détecterait des variations sur toute l’image constamment).

VI- Perspectives d’amélioration (structure globale)

Nous sommes parvenus, sur toute la durée de ce projet, à développer ce qui représente le cœur de l’installation que nous avons imaginée : le programme de traitement de vidéo. Il est fonctionnel et répond au cahier des charges que nous nous étions donné quant à son fonctionnement théorique. Toutefois, il est encore perfectible (gestion de climats extrêmes à développer, entre autres) et le problème de son adaptation aux caméras à module de traitement embarqué reste entier.

Ensuite, il reste tout le reste de l’organisme de notre solution à concevoir : même si nous avons déjà étudié les questions du positionnement des caméras, du câblage à réaliser pour les faire marcher et de la signalisation au niveau du poste de sécurité, rien n’a été précisément défini et médité.

Le développement de la structure globale amènera sans doute d’autres problèmes ou réglages sur lesquels se pencher (par exemple, le seuillage risque fort de se baser sur des caractéristiques différentes entre les caméras intérieures et extérieures).

Il faudrait définir les zones à masquer pour des raisons légales sur toutes les caméras extérieures, ce qui ne pourra être fait qu’après installation. De même, il faudrait étudier l’installation actuelle du poste de sécurité pour en déduire une nouvelle interface humain-machine, soit en adaptant l’existant, soit en en recréant une de toutes pièces.

Conclusion

Tout d'abord, concernant le rendu du projet, nous pouvons être contents des réussites que nous avons apportées à la demande du client. En effet, ce-dernier nous avait demandé d'imaginer un réseau de caméras, ce qui est chose faite. Nous avons aussi estimé le coup de l'installation et avons étudié l'aspect juridique. Il nous était aussi demandé d'écrire un programme de reconnaissance de scènes "suspectes". Nous avons réussi à écrire un programme qui traite un flux vidéo et qui rempli cette tâche. Cependant il ne constitue que le cœur puisqu'il aurait fallu l'implanter sur les caméras. Ce n'est pas faute d'avoir essayé pourtant, nous avons commandé une caméra avec un système d'exploitation intégré afin d'implémenter le programme dans la caméra... tout ce que nous avons réussi à faire fut de recevoir le flux vidéo de cette caméra dans une résolution inexploitable. C'est pourquoi nous nous sommes rabattus sur une webcam, ce qui est certes moins contraignant mais le résultat est quand même au rendez-vous.

La plupart des objectifs ont été atteints, le plus gros échec restant l'impossibilité de faire quelque chose de la caméra mv-BlueLynx.

D’autre part, concernant le travail en équipe, c'est dans une bonne ambiance que nous avons travaillé ensemble. Chacun effectuait le travail qui lui était attribué de manière efficace, personne ne marchait sur les plates bandes d'un autre et l'ensemble des choses à faire était couvert par l'équipe. Malgré la difficulté à se mettre à la tâche chaque semaine, la motivation des uns forçait la motivation des autres.

C'est pourquoi, chaque membre de l'équipe est sorti grandi de cette expérience. En effet, chacun d'entre nous a appris à travailler en équipe ce qui signifie, attribuer les tâches en fonction des capacités des membres de l'équipe, savoir où trouver l'éventuelle aide à la réalisation, entretenir la relation avec le client et l'éventuel intermédiaire, respecter les autres et savoir pardonner les éventuelles erreurs et/ou moindre implication. Nous serons, à coup sûr, confrontés à ces situations que ce soit dans notre carrière professionnelle, dans notre entourage social ou dans le domaine familial. Il nous reste beaucoup de choses à apprendre dans le travail d'équipe et cette première expérience nous donne envie d'approfondir.

En conclusion, nous sommes devenus une équipe dans laquelle chacun a trouvé sa place, qui a fait ses preuves et qui finalement n'a pas besoin d'un éventuel chef si ce n'est pour la forme officielle. Nous espérons que les prochaines équipes dont nous ferons partie seront comme celle du PT100-CRS si ce n'est mieux.



Bibliographie

Livres :

- PITAS Ioannis, Digital image processing algorithms, Prentice Hall, 1993
- C. GONZALES Rafael; WINTZ Paul , Digital image processing, Addison Wesley, 1977
- LINGRAND Diane, Introduction au traitement d'images, Vuibert, 2008
- BRES Stéphane; LEBOURGEOIS Frank; JOLION Jean-Michel, Traitement et analyse des images numériques, Hermès, 2003

Thèses :

- Ghorayeb Hicham, Conception et mise en œuvre d'algorithmes de vision temps réel pour la vidéo surveillance intelligente, Informatique temps réel, robotique et automatique, Ecole Nationale des Mines (Paris), 2007, 191p
- Conte Donatello, Detection tracking and behaviour analysis of moving people in intelligent video surveillance systems, Informatique, Institut national des sciences appliquées (Lyon), 2006, 98p

Lois :

Vigineo, Vigineo solutions de vidéo surveillance & sécurité : législation[en ligne], disponible sur :
http://www.vigineo.fr/loi_du_21_janvier_1995_relative_a_la_securite.html
http://www.vigineo.fr/decret_du_17_octobre_1996_relatif_a_la_videosurveillance.html

Normes techniques :

Vigineo, Vigineo solutions de vidéo surveillance & sécurité : législation[en ligne], disponible sur :
http://www.vigineo.fr/arrete_du_26_septembre_2006_normes_video-surveillance.html
http://www.vigineo.fr/decret_du_3_aout_2007_normes_videosurveillance.html

Manuel MatrixVision BlueLynx (accès nécessitant la création gratuite d'un compte d'utilisateur) :

Manuel : http://www.matrix-vision.com/manuals/mvBlueLYNX/
Guide pour la création de programmes de traitement :
http://www.matrix-vision.com/manuals/mvBlueLYNX/ProgrammingOwnApplications_page_copying_progs.html#ProgrammingOwnApplications_section_copying_progs
Exemples d'applications :
http://www.matrix-vision.com/manuals/mvBlueLYNX/dateien/mvslib/ExampleApplications_page_0.html

Nous n'avons pas eu besoin d'utiliser nos recherches sur les brevets car les thèses, notamment celle qui traite de la conception et mise en œuvre d'algorithmes de vision temps réel pour la vidéosurveillance intelligente nous a fourni la plupart de nos informations.



Annexes

Programme principal de traitement en langage C

```
#include <stdio.h>
#include <stdlib.h>
#include "Image.h"
#include "diff.h"

int main( )
{
   char NomImOrig[100] = "image.pgm";
   char NomImRef[100] = "ref.pgm";
   char NomImSeg [100] = "seg.pgm";
   char NomImEro [100] = "ero.pgm";
   char NomImDil [100] = "dil.pgm";

  /* *************************************** */
  /* PHASE 1 - LECTURE de l'IMAGE ORIGINALE  */
  /* *************************************** */

  struct Image* ImOrig = ReadPGM (NomImOrig);
  struct Image* ImRef = ReadPGM (NomImRef);

  /* ***************************************** */
  /* PHASE 2 - SEGMENTATION PAR DIFFERENCE   */
  /* ***************************************** */

  struct Image* ImSeg = diff(ImOrig, ImRef);

  /* ************************************** */
  /* PHASE 3 - EROSION                        */
  /* ************************************** */

struct Image* ImEro=AllocImage(ImSeg->Lignes, ImSeg->Colonnes, 0);

int j;

for(j=2*(ImSeg->Colonnes)+2;j<(ImSeg->Lignes-2)*ImSeg->Colonnes-1;j++)
          {
          if (
                    (ImSeg->data[j-ImSeg->Colonnes-1]==255) &&
                    (ImSeg->data[j-ImSeg->Colonnes]==255) &&
                    (ImSeg->data[j-ImSeg->Colonnes+1]==255) &&
                    (ImSeg->data[j-1]==255) &&
                    (ImSeg->data[j]==255) &&
                     (ImSeg->data[j+1]==255) &&
                    (ImSeg->data[j+ImSeg->Colonnes-1]==255) &&
                    (ImSeg->data[j+ImSeg->Colonnes]==255) &&
```



```
                (ImSeg->data[j+ImSeg->Colonnes+1]==255))
                {
                ImEro->data[j]=255;
                }
        else ImEro->data[j]=0;
        }

 /* ************************************************/
 /* PHASE 4 - DILATATION DE L'IMAGE  SEGMENTEE  */
 /* ************************************************/


struct Image* ImDil=AllocImage(ImSeg->Lignes, ImSeg->Colonnes, 0);

int k;

for(k=ImEro->Colonnes+1;k<(ImEro->Lignes-2)*ImEro->Colonnes;k++)
        {
        if ((ImEro->data[k-ImEro->Colonnes-1]==255) ||
                (ImEro->data[k-ImEro->Colonnes]==255) ||
                (ImEro->data[k-ImEro->Colonnes+1]==255) ||
                (ImEro->data[k-1]==255) ||
                (ImEro->data[k]==255) ||
                (ImEro->data[k+1]==255) ||
                (ImEro->data[k+ImEro->Colonnes-1]==255) ||
                (ImEro->data[k+ImEro->Colonnes]==255) ||
                (ImEro->data[k+ImEro->Colonnes+1]==255))
                {
                ImDil->data[k]=255;
                }
        else ImDil->data[k]=0;
        }

 /* ****************************************** */
 /* PHASE 5 - ENREG. DE L'IMAGE  SEGMENTEE  */
 /* ****************************************** */

 WritePGM (ImSeg, NomImSeg, "Resultat de Seg" );
 WritePGM (ImEro, NomImEro, "Resultat de Ero" );
 WritePGM (ImDil, NomImDil, "Resultat de Dil" );

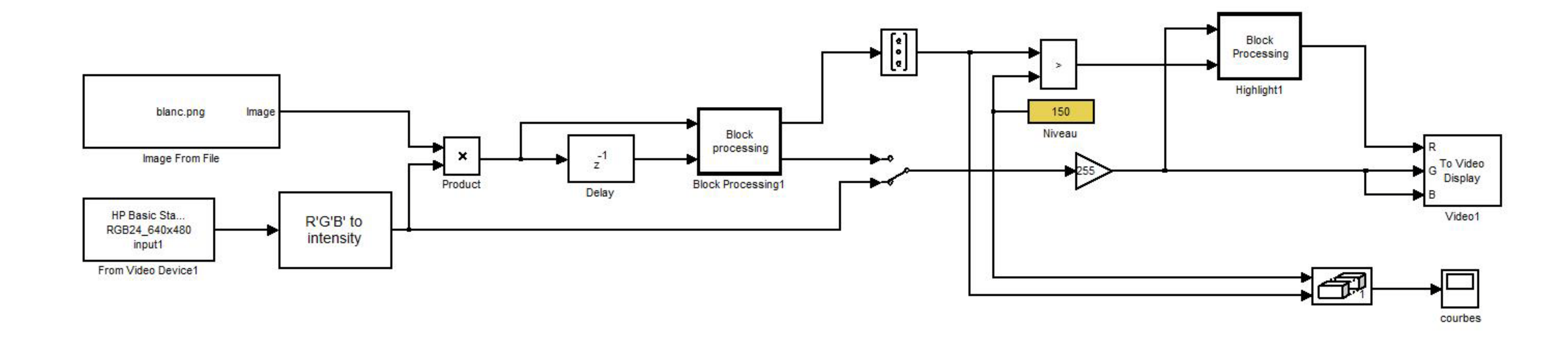
 // Libération de la mémoire
 LibereImage(ImOrig);
 LibereImage(ImSeg);
 LibereImage(ImEro);

 return 0;
}
```



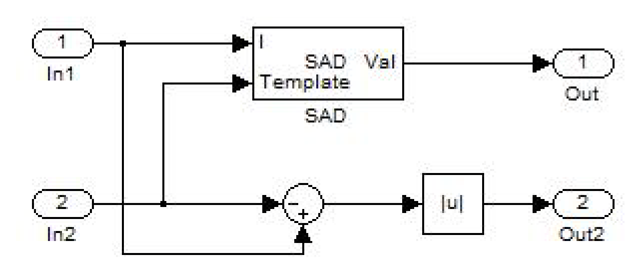
Prototypage sous MATLAB

1. Schéma bloc





2. Contenu du Block ‘processing 1’



3. Contenu du Block ‘Highlight’ (‘processing 2’)

